**EXCEPTION HANDLING (Java) Zeeshan**

Major Exceptions Names  
COMMONLY USE EXCEPTIONS PREVENTS THE APP IN ANDROID :

1.**StackOverflowError** : when your program is in repeatative process and which indicates infinite or deeprecursion when creating nested layouts programatically.like fragments  
2. **NullPointerException** : when compiler not found a instance or a value or anything   
3. **ApplicationNotResponding** : when system killing unresponsive application have hung on a long-running operation  
4. **OutOfMemoryError** :when you are trying to add a large amount of data in lowest memory.  
5.) **InflateException** : This exception is thrown When an error conditions are occurred.

6.) **Surface.OutOfResourceException**: This exception is thrown When a surface is not created or resized.

7.) **SurfaceHolder.BadSurfaceTypeException**: This exception is thrown from the lockCanvas()(Illegal Arjuments) method, when invoked on a Surface whose is SURFACE\_TYPE\_PUSH\_BUFFERS

8.) **WindowManager.BadTokenException**: This exception is thrown at the time of trying to add view an invalid WindowManager.LayoutParamstoken.  
9) **ActivityNotFoundException :** When we are not mention a activity in manifest file.

Following is the list of Java Checked Exceptions Defined in java.lang. in ANDROID

|  |  |
| --- | --- |
| **Sr.No.** | **Exception & Description** |
| 1 | **ClassNotFoundException**  Class not found. |
| 2 | **CloneNotSupportedException**  Attempt to clone an object that does not implement the Cloneable interface. |
| 3 | **IllegalAccessException**  Access to a class is denied. |
| 4 | **InstantiationException**  Attempt to create an object of an abstract class or interface. |
| 5 | **InterruptedException**  One thread has been interrupted by another thread. |
| 6 | **NoSuchFieldException**  A requested field does not exist. |
| 7 | **NoSuchMethodException**  A requested method does not exist. |

Commonly Use of Java Exception Handling Names:

|  |  |
| --- | --- |
| 1 | **ArithmeticException**  Arithmetic error, such as divide-by-zero. |
| 2 | **ArrayIndexOutOfBoundsException**  Array index is out-of-bounds. |
| 3 | **ArrayStoreException**  Assignment to an array element of an incompatible type. |
| 4 | **ClassCastException**  Invalid cast. |
| 5 | **IllegalArgumentException**  Illegal argument used to invoke a method. |
| 6 | **IllegalMonitorStateException**  Illegal monitor operation, such as waiting on an unlocked thread. |
| 7 | **IllegalStateException**  Environment or application is in incorrect state. |
| 8 | **IllegalThreadStateException**  Requested operation not compatible with the current thread state. |
| 9 | **IndexOutOfBoundsException**  Some type of index is out-of-bounds. |
| 10 | **NegativeArraySizeException**  Array created with a negative size. |
| 11 | **NullPointerException**  Invalid use of a null reference. |
| 12 | **NumberFormatException**  Invalid conversion of a string to a numeric format. |
| 13 | **SecurityException**  Attempt to violate security. |
| 14 | **StringIndexOutOfBounds**  Attempt to index outside the bounds of a string. |
| 15 | **UnsupportedOperationException**  An unsupported operation was encountered. |

**EXCEPTION DESCRIPTION:**

An **exception**is a problem that occurs during program execution. Exceptions cause abnormal termination of the program.  
Exceptions can be caught using a combination of the **try**and **catch**keywords.  
A try/catch block is placed around the code that might generate an exception.   
**Types of Exceptions**

Checked and Unchecked Exception. Also called runtime. The main difference is that checked exceptions are checked when compiled, while unchecked exceptions are checked at runtime.  Thread. Sleep() throws an Interrupted Exception. This is an example of a **checked**exception. Your code will not compile until you've handled the exception.

ArrayIndexOutOfBoundsException:  
public class MyClass {  
public static void main(String[ ] args) {  
**try** {  
int a[ ] = new int[2];  
System.out.println(a[5]);  
} **catch** (Exception e) {  
System.out.println("An error occurred");  
}  
}  
}  
//Outputs "An error occurred"

**Briefly Describe the exception use.**

e.printStackTrace();

ArithmeticException (Customized with Throw)

**package** Exception;

**public** **class** myException

{

**public** **static** **void** main(String[] args) {

**int** c=myException.*div*(3,0);

System.***out***.println(c);

}

**static** **int** div(**int** a,**int** b) **throws** ArithmeticException {

**if**(b == 0) {

**throw** **new** ArithmeticException("Division by Zero does not containt");

} **else** {

**return** a / b;

}

}

}

**Application Not Responding (ANR)**You can’t exactly handle the ANR.  
To be precise ANR is a condition  
**it is not an “Exception” or “Error”**  
Android is very clear about it that it’s a dialog. They never call it Exception or Error.

When application UI is not responding then we will apply the threads.  
Our program can make optimal use of available resources by running two or more components concurrently, with each component handling a different task.  
You can subdivide specific operations within a single application into individual **threads**that all run in parallel.

Fact is, we can not catch it. but we could avoid this issue not to happen by using background process like AsynTask or Threads and Handler.

How to handle (ANR): Video-LINK **https://youtu.be/kpFwxJFYnOo**

Throws keyword define the type of exception method  
throw keyword customized your exception message

A single try block can contain multiple catch blocks that handle different exceptions separately.  
**Example:**try {  
//some code  
} **catch** (ExceptionType1 e1) {  
//Catch block  
} **catch** (ExceptionType2 e2) {  
//Catch block  
} **catch** (ExceptionType3 e3) {  
//Catch block  
}